

|  |
| --- |
|  |

Switch Abstraction Interface

Change Proposal

|  |  |
| --- | --- |
| **Title** | **New API for xSTP** |
| **Authors** | **DELL** |
| **Status** | **In Review** |
| **Type** | **Standards Track** |
| **Created** | **03/04** |
| **SAI-Version** | **V0.9.2** |

**Contents**

[List of Changes i](#_Toc416073813)

[1 Overview 1](#_Toc416073814)

[2 Specification 1](#_Toc416073815)

[2.1 Create STP instance 2](#_Toc416073816)

[2.2 Delete STP instance 2](#_Toc416073817)

[2.3 Update stp state on stp instance of a port 3](#_Toc416073818)

[2.4 Retrieve stp state on stp instance of a port 3](#_Toc416073819)

[2.5 Get/Set Attribute for XSTP instance 4](#_Toc416073820)

[2.6 Method Table 5](#_Toc416073821)

[2.7 Removing existing STP related Attribute from Port API List 5](#_Toc416073822)

[2.8 Associating a VLAN to STP instance 6](#_Toc416073823)

[2.9 Retrieve list of VLANs mapped for a STP instance 6](#_Toc416073824)

[3 Examples 6](#_Toc416073825)

[3.1 Create STP instance 6](#_Toc416073826)

[3.2 Get all the VLANs in the default STG instance 6](#_Toc416073827)

[3.3 Associate Vlan to a STP instance 7](#_Toc416073828)

[3.4 Assign STP port state 8](#_Toc416073829)

[3.5 Get STP port state 8](#_Toc416073830)

[3.6 Destroy the STG instance when Vlans are associated 8](#_Toc416073831)

[3.7 Set attribute for the STG instance 9](#_Toc416073832)

[3.8 Disassociate Vlan from a STP instance and destroy the STG instance 9](#_Toc416073833)

[3.9 Get attribute for the invalid STG instance 10](#_Toc416073834)

[4 Appendix 10](#_Toc416073835)

# List of Changes

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Changes | Name | Date |
| 0.9.2 | Proposal for xSTP | 0 | 03/04/2015 |
| 0.9.2 | Version 2   * Incorporated feedbacks. Appendix section has details of the changes |  | 03/05/2015 |
| 0.9.2 | Version 3   * Changed according to the unified object id model. * Added SAI\_STP\_ATTR\_VLAN as an READ-ONLY attribute |  | 03/17/2015 |
| 0.9.2 | Version 4   * Changed overview and added examples * Added sai\_vlan\_list\_t in sai\_attribute\_value\_t |  | 04/07/2015 |

License

© 2014 Microsoft Corporation, Dell Inc., Facebook, Inc, Broadcom Corporation, Intel Corporation, Mellanox Technologies Ltd.

As of September 9, 2014, the following persons or entities have made this Specification available under the Open Web Foundation Final Specification Agreement (OWFa 1.0), which is available at <http://www.openwebfoundation.org/legal/the-owf-1-0-agreements/owfa-1-0>

Microsoft Corporation, Dell Inc., Facebook, Inc, Intel Corporation, Mellanox Technologies Ltd.

You can review the signed copies of the Open Web Foundation Agreement Version 1.0 for this Specification at <http://opencompute.org/licensing/>, which may also include additional parties to those listed above.

Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, noninfringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE FOLLOWING IS A LIST OF MERELY REFERENCED TECHNOLOGY: Microprocessor technology, semiconductor manufacturing technology, operating system technology (including without limitation networking operating system technology), emulation technology, graphics technology, video technology, integrated circuit packaging technology and the like, compiler technologies, object oriented technology, optical/RF communications technology including chip I/O and driver technology, bus technology, memory chip technology (including, without limitation, NAND memory, NOR memory, resistive RAM (RRAM), seek scan probe (SSP) memory, nonvolatile memory (including without limitation, memory based on chalcogenide materials, phase change memory (PCM), one or more stacked layers of memory cells, embedded PCM memories, non-volatile cache memory, solid state drives, SRAM, embedded DRAM, ferro-electric memory, and polymer memory)) and/or health-related and medical technology. IMPLEMENTATION OF THESE TECHNOLOGIES MAY BE SUBJECT TO THEIR OWN LEGAL TERMS.

# Overview

This document describes the APIs to create, remove and update Spanning-Tree Protocol (STP) instances. User can use the create function call to create a STP instance and then associate a VLAN or multiple VLANs to this STP instance. User can also set the Spanning-Tree port state for the STP instance.

A default STP instance is created during SAI initialization phase. User can use SAI\_SWITCH\_ATTR\_DEFAULT\_STP\_INST\_ID to query the default STP instance.

* When a VLAN is created, it is associated to the default STP instance by default.
* User can associate the VLAN to a non-default STP instance, and the VLAN is removed from the default STP instance.
* A STP instance is not allowed to be deleted if it has VLAN’s associated. The user has to associate the VLAN’s to a new STP instance or default STP instance before deleting the STP instance
* When a VLAN associated with a non-default STP instance is removed, it will also be removed from the STP instance

# Specification

**In sai.h**

typedef enum \_sai\_api\_t {

SAI\_API\_STP= 18, /\* sai\_stp\_api\_t \*/

}

**In saistatus.h**

/\*

\* Invalid STP Instance ID

\*/

#define SAI\_STATUS\_INVALID\_STP\_INSTANCE SAI\_STATUS\_CODE(0x00000018L)

**In saiswitch.h**

typedef enum \_sai\_switch\_attr\_t {

/\*READ - ONLY. Default SAI STP instance ID \*/

SAI\_SWITCH\_ATTR\_DEFAULT\_STP\_INST\_ID,

} sai\_switch\_attr\_t

**In saitypes.h**

typedef struct \_sai\_vlan\_list\_t {

uint32\_t vlan\_count;

sai\_vlan\_id\_t \*vlan\_list;

} sai\_vlan\_list\_t;

typedef union {

…..

sai\_vlan\_list\_t vlanlist

}sai\_attribute\_value\_t;

**These changes are in new file saistp.h**

typedef enum \_sai\_stp\_attr\_t {

/\* READ - ONLY \*/

SAI\_STP\_ATTR\_VLAN,

} sai\_stp\_attr\_t

## Create STP instance

/\*  
\* Routine Description:  
\*    Create a STP instance  
\*  
\* Arguments:  
\*    [out] inst\_id - Spanning tree instance identifier  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_create\_stp\_inst\_fn)  
  (  
     \_Out\_ sai\_object\_id\_t \*inst\_id,     /\* instance id\*/  
  )

## Delete STP instance

/\*  
\* Routine Description:  
\*    Delete STP instance  
\*  
\* Arguments:  
\*    [in] inst\_id - STP instance identifier  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_remove\_stp\_inst\_fn)  
  (  
     \_In\_ sai\_object\_id\_t inst\_id,     /\* instance id\*/  
  )

## Update stp state on stp instance of a port

/\*  
\* Routine Description:  
\*    Update stp state of a port in specified stp instance.    
\*  
\* Arguments:  
\*    [in] inst\_id - STP instance identifier  
\*    [in] port\_id - port identifier  
\*    [in] stp\_port\_state - STP state  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_set\_stp\_port\_state\_fn)  
 (  
     \_In\_ sai\_object\_id\_t inst\_id,     /\* instance id\*/  
     \_In\_ sai\_object\_id\_t port\_id,   /\*port identifier\*/  
     \_In\_ sai\_port\_stp\_port\_state\_t  stp\_port\_state /\*STP state\*/  
 )

## Retrieve stp state on stp instance of a port

/\*  
\* Routine Description:  
\*    Get stp state of a port in specified stp instance.    
\*  
\* Arguments:  
\*    [in] inst\_id - STP instance identifier  
\*    [in] port\_id - port identifier  
\*    [out] stp\_port\_state - STP state  
\*  
\* Return Values:  
\*    SAI\_STATUS\_SUCCESS on success  
\*    Failure status code on error  
\*/

typedef sai\_status\_t (\*sai\_get\_stp\_port\_state\_fn)  
  (  
     \_In\_ sai\_object\_id\_t inst\_id,     /\* instance id\*/  
     \_In\_ sai\_object\_id\_t port\_id,   /\*port identifier\*/  
     \_Out\_ sai\_port\_stp\_port\_state\_t  \*stp\_port\_state /\*STP state\*/  
  )

## Get/Set Attribute for XSTP instance

/\*

\* Routine Description:

\* Set STP attribute

\*

\* Arguments:

\* [in] sai\_object\_id\_t– stp id

\* [in] attr - attribute

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_set\_stp\_attribute\_fn)(

\_In\_ sai\_object\_id\_t stp\_id,

\_In\_ const sai\_attribute\_t \*attr

);

/\*

\* Routine Description:

\* Get STP attribute

\*

\* Arguments:

\* [in] sai\_object\_id\_t– stp id

\* [in] attr\_count - number of attributes

\* [inout] attr\_list - array of attributes

\*

\* Return Values:

\* SAI\_STATUS\_SUCCESS on success

\* Failure status code on error

\*/

typedef sai\_status\_t (\*sai\_get\_stp\_attribute\_fn)(

\_In\_ sai\_object\_id\_t stp\_id,

\_In\_ uint32\_t attr\_count,

\_Inout\_ sai\_attribute\_t \*attr\_list

);

## Method Table

/\* \* STP methods table retrieved with sai\_api\_query() \*/

typedef struct \_sai\_xstp\_api\_t {

sai\_create\_stp\_inst\_fn  create\_stp\_inst;

sai\_remove\_stp\_inst\_fn remove\_stp\_inst;

sai\_set\_stp\_port\_state\_fn set\_stp\_port\_state;

sai\_get\_stp\_port\_state\_fn get\_stp\_port\_state;

sai\_set\_stp\_attribute\_fn set\_stp\_attribute;

sai\_get\_stp\_attribute\_fn get\_stp\_attribute;

} sai\_stp\_api\_t;

## Removing existing STP related Attribute from Port API List

**In saiport.h**

The below attribute needs to be removed from the list of port attributes

/\* Stp mode [sai\_port\_stp\_state\_t] \*/

SAI\_PORT\_ATTR\_STP\_STATE,

**The below definition of STP States needs to be moved from saiport.h to this the saixstp.h file**

/\*

\* Attribute data for SAI\_PORT\_STP\_STATE

\*/

typedef enum \_sai\_port\_stp\_state\_t

{

/\* Port is Learning \*/

SAI\_PORT\_STP\_STATE\_LEARNING,

/\* Port is Forwarding \*/

SAI\_PORT\_STP\_STATE\_FORWARDING,

/\* Port is Blocking \*/

SAI\_PORT\_STP\_STATE\_BLOCKING,

} sai\_port\_stp\_state\_t;

## Associating a VLAN to STP instance

**In saivlan.h**

Add an attribute in existing VLAN Module to associate a STP instance with a VLAN ID.

typedef enum \_sai\_vlan\_attr\_t

{

/\* Maximum number of learned MAC addresses [uint32\_t] \*/

SAI\_VLAN\_ATTR\_MAX\_LEARNED\_ADDRESSES,

/\* STP Instance that the VLAN is associated to[sai\_object\_id\_t]\*/

SAI\_VLAN\_ATTR\_STP\_INSTANCE,

/\* Custom range base value \*/

SAI\_VLAN\_ATTR\_CUSTOM\_RANGE\_BASE = 0x10000000

} sai\_vlan\_attr\_t;

## Retrieve list of VLANs mapped for a STP instance

Use SAI\_STP\_ATTR\_VLAN attribute. Use vlan\_data of type sai\_vlan\_list\_t to access the attribute

# Examples

## Create STP instance

sai\_api\_query(SAI\_API\_STP, &sai\_stp\_api);

sai\_object\_id stp\_object\_id;

sai\_stp\_api-> create\_stp\_inst(&stp\_object\_id);

## Get all the VLANs in the default STG instance

/\* Retrieve the Switch API method and get the default STG instance id \*/

sai\_api\_query(SAI\_API\_SWITCH, &sai\_switch\_api);

sai\_attribute\_t get\_switch\_attr;

get\_switch\_attr.id = SAI\_SWITCH\_ATTR\_DEFAULT\_STP\_INST\_ID

sai\_switch\_api-> get\_switch\_attribute(1,& get\_switch\_attr);

/\* Get the vlans associated to default STG instance id \*/

sai\_attribute\_t get\_stp\_attr\_list;

//Initialize the members of get\_stp\_attr\_list;

get\_stp\_attr\_list.id = SAI\_STP\_ATTR\_VLAN;

get\_stp\_attr\_list.value.vlanlist.vlan\_count = 10;

get\_stp\_attr\_list.value.vlanlist.vlan\_list = calloc (10, sizeof(sai\_vlan\_id\_t));

sai\_stp\_api-> get\_stp\_attribute (get\_switch\_attr.value.u64, 1, &get\_attr\_list);

//Check if SAI\_STATUS\_BUFFER\_OVERFLOW is returned

Now Invoke the get\_stp\_attribute api with the get\_stp\_attr\_list.value.vlanlist.vlan\_count number of memory allocated.

get\_stp\_attr\_list.value.vlanlist.vlan\_list = calloc (get\_stp\_attr\_list.value.vlanlist.vlan\_count, sizeof(sai\_vlan\_id\_t));

sai\_stp\_api-> get\_stp\_attribute (get\_switch\_attr.value.u64, 1, &get\_attr\_list);

//Check if SAI\_STATUS\_SUCCESS is returned

//Check if all the vlans from 1 to 4094 are part of default STG

## Associate Vlan to a STP instance

sai\_api\_query(SAI\_API\_VLAN, &sai\_vlan\_api);

sai\_vlan\_id\_t vlan\_id = 2;

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

sai\_vlan\_api->sai\_create\_vlan(vlan\_id);

set\_attr.id = SAI\_VLAN\_ATTR\_STP\_INSTANCE;

set\_attr.value.u64 = stp\_object\_id;

sai\_vlan\_api-> set\_vlan\_attribute (vlan\_id, &set\_attr);

get\_attr.id = SAI\_VLAN\_ATTR\_STP\_INSTANCE;

get\_attr.value.u64 = 0;

sai\_vlan\_api-> set\_vlan\_attribute (vlan\_id, &set\_attr);

Check if the value of get\_attr.value.u64 is stp\_object\_id;

//Now get all the VLANs associated to default STG instance as described in case 2 and verify that vlan 2 should not be associated to default STG.

## Assign STP port state

sai\_object\_id\_t port\_object\_id = 100

sai\_stp\_api-> set\_stp\_port\_state (stp\_object\_id, port\_object\_id, SAI\_PORT\_STP\_STATE\_LEARNING);

port\_object\_id = 200

sai\_stp\_api-> set\_stp\_port\_state (stp\_object\_id, port\_object\_id, SAI\_PORT\_STP\_STATE\_BLOCKING);

port\_object\_id = 300

sai\_stp\_api-> set\_stp\_port\_state (stp\_object\_id, port\_object\_id, SAI\_PORT\_STP\_STATE\_FORWARDING);

## Get STP port state

sai\_object\_id\_t port\_object\_id = 300

sai\_port\_stp\_state\_t stp\_port\_state = 0;

sai\_stp\_api-> get\_stp\_port\_state (stp\_object\_id, port\_object\_id, &stp\_port\_state);

Check if the stp\_port\_state is SAI\_PORT\_STP\_STATE\_FORWARDING

Sai\_object\_id\_t port\_object\_id = 200

sai\_port\_stp\_state\_t stp\_port\_state = 0;

sai\_stp\_api-> get\_stp\_port\_state (stp\_object\_id, port\_object\_id, &stp\_port\_state);

Check if the stp\_port\_state is SAI\_PORT\_STP\_STATE\_BLOCKING

## Destroy the STG instance when Vlans are associated

sai\_stp\_api-> remove\_stp\_inst (stp\_object\_id);

//Check if SAI\_STATUS\_ATTR\_OBJECT\_IN\_USE is returned since Vlans are still associated to this STG instance

## Set attribute for the STG instance

sai\_attribute\_t set\_attr;

set\_attr\_id = SAI\_STP\_ATTR\_VLAN;

sai\_stp\_api-> set\_stp\_attribute (stp\_object\_id, &set\_attr);

//Check if SAI\_STATUS\_ATTR\_NOT\_SUPPORTED\_0 is returned.

## Disassociate Vlan from a STP instance and destroy the STG instance

/\* Get the default STG instance id \*/

sai\_api\_query(SAI\_API\_SWITCH, &sai\_switch\_api);

sai\_attribute\_t get\_switch\_attr;

get\_switch\_attr.id = SAI\_SWITCH\_ATTR\_DEFAULT\_STP\_INST\_ID

sai\_switch\_api-> get\_switch\_attribute(1,& get\_switch\_attr);

sai\_api\_query(SAI\_API\_VLAN, &sai\_vlan\_api);

sai\_attribute\_t set\_attr;

sai\_attribute\_t get\_attr;

Initialize set\_attr and get\_attr.

sai\_vlan\_api->sai\_create\_vlan(vlan\_id);

set\_attr.id = SAI\_VLAN\_ATTR\_STP\_INSTANCE;

set\_attr.value.u64 = get\_switch\_attr.value.u64;

sai\_vlan\_api-> set\_vlan\_attribute (vlan\_id, &set\_attr);

get\_attr.id = SAI\_VLAN\_ATTR\_STP\_INSTANCE;

sai\_vlan\_api-> get\_vlan\_attribute (vlan\_id, &set\_attr);

Check if the value of get\_attr.value.u64 is default stg instance object id;

//Now get all the VLANs associated to default STG instance as described in case 2 and verify that vlan 2 should now be associated to default STG.

sai\_stp\_api-> remove\_stp\_inst (stp\_object\_id);

//Now the remove should be successful.

## Get attribute for the invalid STG instance

sai\_attribute\_t get\_attr;

set\_attr\_id = SAI\_STP\_ATTR\_VLAN;

sai\_stp\_api-> get\_stp\_attribute (0xFFFFFFFFF, 1, &get\_attr);

//Check if SAI\_STATUS\_INVALID\_STP\_INSTANCE is returned.

# Appendix

**Review Comments – Atit 03/05/2015**

From: Jain, Atit [<mailto:Atit.Jain@caviumnetworks.com>]   
Sent: Thursday, March 05, 2015 6:33 PM  
To: Manickam, Arunsubash; [opencompute-networking@lists.opencompute.org](mailto:opencompute-networking@lists.opencompute.org)  
Subject: RE: API Proposal for STG

Hi Arun,

The SAI\_STP\_ DEFAULT\_INST\_ID should be part of SAI Switch attributes because its per NPU/Switch default. In its current form, SAI STP this as an attribute is set for a sai\_object\_id\_tstp\_id which is like setting default instance per instance.

Few minor comments are in the file attached.

Regards,  
Atit

From: [opencompute-networking-bounces@lists.opencompute.org](mailto:opencompute-networking-bounces@lists.opencompute.org) [<mailto:opencompute-networking-bounces@lists.opencompute.org>] On Behalf Of Manickam, Arunsubash  
Sent: Thursday, March 05, 2015 6:43 PM  
To: [Atit.Jain@caviumnetworks.com](mailto:Atit.Jain@caviumnetworks.com); [opencompute-networking@lists.opencompute.org](mailto:opencompute-networking@lists.opencompute.org)  
Subject: Re: [Opencompute-networking] API Proposal for STG

Hi Atit,

Thanks for your quick feedback

1 – I agree that SAI\_STP\_DEFAULT\_INST\_ID should be part of switch attribute. This would be inline with default VLAN ID which is a switch attribute - SAI\_SWITCH\_ATTR\_DEFAULT\_PORT\_VLAN\_ID

2 – Though we will have no attributes defined we will have the get/set attribute functions defined

3 - SAI\_STATUS\_INVALID\_STP\_INSTANCE – Yes we will add this new error code to return failure for an invalid instance passed

We will update this and send an updated proposal